

sequenor helper文档

sequenor helper是一个用于封装和管理由Java Block Sequencor这一bb插件的数据包和python脚本工具。Java Block Sequencor可以生成帧动画模型，而sequenor helper可以方便地自动化生成对应的物品模型映射文件，并使用简洁统一的参数调用和播放动画。

文件结构

sequenor helper包含两个部分：用于物品模型映射生成的python脚本，和用于播放动画的数据包。

物品模型映射生成

生成脚本`generate.py`专为Vscode设计，你需要打开帧动画资源包所在的文件夹，并安装Python3和对应的vsc插件，确保vscode可以执行python脚本。

首先你需要使用Java Block Sequencor制作和导出帧动画模型，并将所有的动画放在同一个母目录下，保证其为模型目录/动画目录/帧.json，并且注意指定好纹理位置；

然后将生成脚本放入资源包目录，以资源包根目录为执行位置，并指定下列的参数，这里给出了一个示例。

```
"""
Minecraft帧动画序列模型映射文件生成脚本

参数说明：
- 源路径SOURCE_PATH：指定帧动画的目录，为一相对路径，格式必定为<任意路径>/assets/<命名空间>/models/<其他路径>
- 目标路径TARGET_PATH：生成模型映射文件的路径，相对路径（不需要带./）
- 文件名FILE_NAME：映射文件的名字，程序会自动添加.json后缀
- 动画列表ANIMATION_LIST：字符串列表，每一个项代表一个动画，表示原路径下的一个文件夹，内部为按数字标识的帧动画模型文件
- 回落映射FALLBACK_MAPPING：可选，字符串，表示在不播放动画时的模型；若不存在则使用动画列表第一项的第一个模型
- 动画索引ANIMATION_INDEX：影响动画索引字符串在物品custom_model_data.strings的第几项，例如0表示决定模型播放哪个动画的字符串在对应cmd的第一项
- 帧索引FRAME_INDEX：影响帧索引字符串在物品custom_model_data.floats的第几项
"""

# 输入参数配置区 - 修改这里的参数来生成不同的映射文件
SOURCE_PATH = "assets\kaleidoscope_lab\models\leave_me_alone_box" # 源路径：指定帧动画的目录
TARGET_PATH = "assets\kaleidoscope_lab\items" # 目标路径：生成模型映射文件的路径
FILE_NAME = "leave_me_alone_box" # 文件名：映射文件的名字（不含.json后缀）
ANIMATION_LIST = ["open", "close1", "close2", "close3", "close4"] # 动画列表：每个项代表一个动画
FALLBACK_MAPPING = None # 回落映射：可选，表示在不播放动画时的模型
ANIMATION_INDEX = 0 # 动画索引：影响最外层index
FRAME_INDEX = 0 # 帧索引：影响内层帧对应的index
```

完成参数填写并执行后即可在对应目标路径下找到生成的物品模型映射文件。

数据包管理

数据包用于播放导出的帧动画模型映射。数据包提供了三个api：`sh:init`、`sh:start`和`sh:end`。其余函数为内部函数，使用者不需要了解。

使用该数据包时，使用者需要首先生成引用了上述模型的物品展示实体，并且确保引用到的cmd序号存在。以展示实体为执行者，执行参数配置和开始播放指令：

```
#初始化所有参数
function sh:init

#在指定的storage指定参数
data modify storage sh:props data merge value
{id:"open",frames:5,type:2,anim_index:0,frame_index:0,function:"kaleidoscope:lab/trans/leave_me_alone_box/roll_close"}

#执行开始动画函数
function sh:start
```

`sh:init`函数用于初始化所有动画相关参数；

指定的参数位于`sh:props data`这一storage中，使用者需要使用`data modify ... merge ...`指令向其中合并参数；

`sh:start`函数用于在指定完参数后按照参数执行动画播放行为。

参数及释义

所有可能的参数及其类型和解释如下：

- `id(string)`: 字符串，表示本次播放的动画，对应ANIMATION_LIST中的一项；
- `frames(int)`: 表示该动画的总帧数，从0开始计数，和插件导出的帧动画编号一致；
- `type(enum int)`: 播完动画后执行的操作类型：
 - 1表示循环，重置回第一帧重新播放；
 - 2表示简单停止，保留所有参数并停留在最后一帧；
 - 其他数值表示重置，清除所有参数并回到回落映射代表的模型。
- `anim_index(int)`: 动画索引，同ANIMATION_INDEX；
- `frame_index(int)`: 帧索引，同FRAME_INDEX；
- `function(string)`: 可选，回调函数，在动画播放完成后，在模型所在位置以模型为执行者执行的一个函数。
 - 可以为宏函数，参数直接填写在函数路径后面，其中双引号需要转义。
 - 如：`{function:"foo:bar with entity @s data"}`

其他api信息

- 函数api：
 - `sh:init`函数用于初始化所有动画相关参数；
 - `sh:start`函数用于在指定完参数后按照参数执行动画播放行为。
 - `sh:end`函数在动画播放完成后自动调用，也可以在动画中途调用，清除所有参数并返回回落映射。
- 只读数据：

- 注：数据包的机制依赖这些数据运行！除非你确信自己的操作所造成的后果，否则不要随意更改这些值！
- `sh_frame`计分板：表示模型当前播放的帧，可以读取该数值并进行播放音效等操作；
- `sh_max_frame`计分板：表示当前动画的总帧数；
- `sh_type`计分板：表示动画播放结束后执行的操作，对应参数中的`type`；
- 实体的`data.sh_animation`存储：传入的参数存储在此处，数据结构和`storage sh:props data`完全相同，可在此读取动画id等。